Application Explorations for Future Interconnects

R.F. Barrett, C.T. Vaughan, and S.D. Hammond
Sandia National Laboratories
Albuquerque, NM, USA
{rfbarre,ctvaugh,sdhammo}@sandia.gov

D. Roweth
Cray, Inc.
Reading, UK
droweth@cray.com

***Abstract*—For over two decades the dominant means for enabling portable performance of computational science and engineering applications on parallel processing architectures has been the bulk-synchronous parallel programming model. Code developers, motivated by performance considerations to minimize the number of messages transmitted, have typically strived to increase the size of each message through aggregation strategies. Emerging and future architectures, especially those seen as targeting Exascale capabilities, provide motivation and capabilities for revisiting this approach. In this paper we explore alternative configurations within the context of a large-scale complex multi-physics application and a proxy that represents its behavior, presenting results that demonstrate some important advantages as the number of processors increases in scale.**

***Index Terms*—High performance computing; parallel architectures; computational science and engineering.**

I. Introduction

For more than two decades, the bulk-synchronous parallel programming model (BSP) [22] has been the dominant theoretical computational model for the implementation of large scale, high-performance computation science and engineering (CSE) applications. This has been aided in part by the standardization of the Message-Passing interface (MPI) which has provided a stable, portable and a largely performant runtime environment for applications written to the BSP paradigm.

As widely available parallel processing architectures have evolved to focus on increasing interconnect bandwidth (relative to latency), application developers have optimized their code to "bulk up" inter-process communication – essentially choosing to aggregate data from various structures into fewer, single, larger messages [5]. Although many applications have continued to perform well up to petascale [2], [8] using a bulk-message approach, for a variety of reasons, including hardware design limitations, increasing interconnect power consumption and changes in software design, this situation is expected to change in future machines [1], [21].

Our work is motivated and driven by our experience in running large-scale applications serving mission critical functions across a breath of agencies including the United States Departments of Energy and Defense. These experiences include empirical characteristics observed in running many applications at full machine scale where artifacts such as messaging rate and interconnect bandwidth can create significantly different behaviors than seen at smaller runs [23]. The contribution of the work described in this paper is to highlight the need to revisit traditional application configuration strategies. Specifically, we illustrate the benefits of careful attention to logical-to-physical mappings of parallel processes as a function of network topologies, and then demonstrate the effectiveness of a message passing strategy that greatly reduces the need for message aggregation. We note that the outcomes presented are at odds with the conventional approaches to communication optimization performed over the past two decades but are in some sense a sentinel for the strategies to come as machine architectures are developed for Exascale-class computing.

This paper is organized as follows. After a brief discussion of the BSP model and related work, we describe the platforms and methodology for our experiments. Next we describe the experiments, progressing from the current implementation of a representative well-known CSE application through some reconfigurations using a proxy application that enables rapid exploration of alternative configurations. We conclude with a summary of our findings and a discussion of future work.

A. The Bulk Synchronous Parallel (BSP) Model

The BSP programming model is the predominant "bridging model" in parallel computing for science and engineering applications. It makes a clear distinction between computation and communication, providing a clear distinction between hardware and software. This approach has enabled continued and improving performance of parallel applications in spite of the rapid evolution of architectures and enabling technologies.

By *message aggregation* we mean the gathering of data from different computational regions, from (potentially) non-contiguous memory locations, into a single user-managed message buffer. The receiving process must then scatter the data out into (again, potentially) non-contiguous data locations, typically before computation can again proceed. This reduces the number of inter-process communication steps, exploiting the node interconnect bandwidth capabilities, while simultaneously avoiding message latencies. We abbreviate this BSP supplemented with message aggregation as BSPMA.

The BSPMA model incurs or exacerbates four costs, none of which advances the computation: consumption of memory (the message buffers), on-node bandwidth (copying into and out of the buffers), synchronization (once the data is transferred), and interference with caching of the computation (and potentially data transfer information). Further, from an application development viewpoint, it interferes with the natural mapping of algorithms to programming languages. BSPMA also typically induces intermediate data buffering requirements, subverting the node interconnect capabilities designed specifically to reduce the communication costs, inhibits asynchronous movement of data, and interferes with the networks' ability to interleave data across the topology.

B. Related Work

For a breadth of CSE application programs, inter-process communication is captured within higher level interfaces [5]. Our focus herein is on the inter-process communication strategies that we expect to be effective on future architectures. Toward that end, the capabilities of the Cray XE6 are seen to be representative of the sorts of architectural capabilities that might be exploited while minimizing the impact to the huge investment in application codes. Hagar, Jost, and Rabenseifner [12] illustrate the effects of process topologies on the performance of the NAS Parallel Benchmarks [4] configured for OpenMP MPI hybrid on clusters of multi-core SMP nodes. Both of these studies were based on the Cray XT4 and XT5, and an IBM Power5 from Bull. Yu, Chung, and Moreira performed related studies on the BlueGene L architecture [24]. Hoefler et al [15] describe the new topology interface in MPI 2.2, which could provide a portable interface for the processor mapping we found effective.

II. Experimental Platforms

In this study, we employed three high performance computing platforms that span a set of processors, interconnects, and interconnect topologies of interest to the NNSA/ASC program and wider HPC community: Cielo, a Cray XE6, a ASC capability platform; Chama, an Intel/Infiniband capacity cluster, and Cascade, a Cray XC. We selected these machines for their diversity of some important characteristics yet with similarity in other areas that allow for some important interpretations. A summary of some important specifications is shown in Table I.

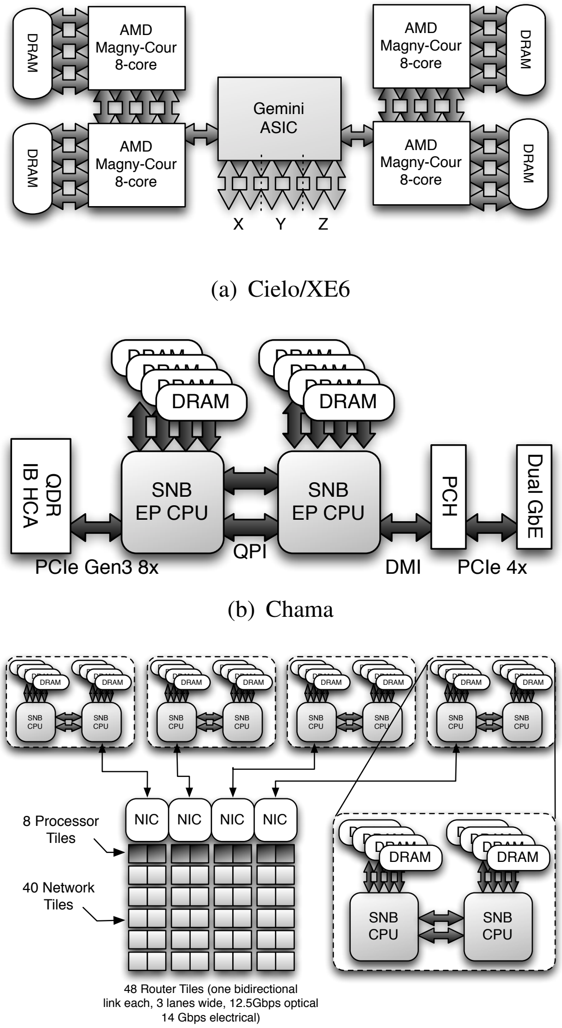
Before providing additional detail on these platforms, we begin with some terminology. Message injection rate is the ability of a node to place messages onto the network interconnect card (NIC). Injection bandwidth measures the ability of the NIC to put data onto the interconnect. Global bandwidth measures the ability of the interconnect to move data between multiple nodes.

Cielo, an instantiation of a Cray XE6, is composed of AMD Opteron Magny-Cours oct-core processors, connected using a Cray custom interconnect named Gemini, and a light-weight kernel operating system called Compute Node Linux (CNL). The system consists of 8,944 dual socket compute nodes, for a total of 143,104 cores. Each processor is divided into two four processor core memory regions, called NUMA nodes (illustrated in Figure 1(a)), connected using HyperTransport version 3.

Nodes are connected using Cray's Gemini 3-D torus interconnect. A Gemini ASIC supports two

Platform	**Cielo**	**Chama**	**Piz Daint**
Processor	AMD Opteron Magny-Cours	Intel Xeon Sandy Bridge	Intel Xeon Sandy Bridge
Nodes	8,518	1,232	2,256
Sockets/node	2	2	2
Cores/socket	8	8	8
Total number of cores	136,288	19,712	36,096
Clock speed (GHz)	2.4	2.6	2.6
Memory per node (GB)	32	32	32
Memory	DDR4 1333 MHz	DDR3 1600 MHz	DDR3 1600 MHz
Socket Connection	HyperTransport	PCIe Gen2 / QPI	PCIe Gen3
Interconnect	Gemini 3D-torus	Qlogic QDR Fat tree	Cray Aries, Dragonfly

TABLE I
KEY ARCHITECTURE PARAMETERS



(a) Cielo/XE6

(b) Chama

(c) Cascade

Fig. 1. Node Architectures

compute nodes. The X and Z dimensions use twice as many links as the Y dimension (24 bits and 12 bits respectively) and therefore introduce an asymmetry to the nodes in terms of bandwidth in the torus. This needs to be taken into account when configuring a system in order to balance the bisection bandwidth of each dimensional slice in the torus. Cielo is configured as a $16 \times 12 \times 24$ 3-D torus. Injection bandwidth is limited by the speed of the Opteron to Gemini HyperTransport link, which runs at 4.4 GT/s. Links in the X and Z dimensions have a peak bi-directional bandwidth of 18.75 GB/s, and the Y dimension peaks at 9.375 GB/s.

Chama, constructed by Appro, Inc., is designed for production capacity computing by the NNSA ASC Trilabs. The system consists of 1,232 compute nodes, connected by a QLogic Infiniband fat tree, using 12000 series switches and 7300 series adapters. Each node is composed of two oct-core Intel Xeon E5-2670 Sandy Bridge processors, illustrated in Figure 1(b), for a total of 19,712 cores, running a RHEL operating system.

Piz Daint, a Cray XC Cascade system, is composed of Intel Xeon Sandy Bridge oct-core processors, connected using a Cray Aries custom interconnect [3] named Dragonfly [18]. The system consists of 2,256 dual socket compute nodes, for a total of 36,096 cores.

III. A REPRESENTATIVE APPLICATION

CTH, used throughout the United States DOE complex, and within the US Department of Defense's (DoD) High Performance Computing Modernization Program (HPCMP) [9], is a multi-material,

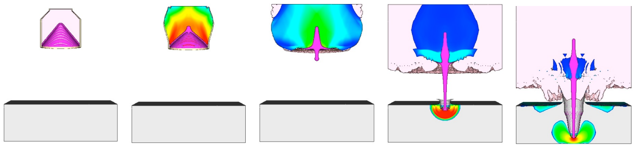
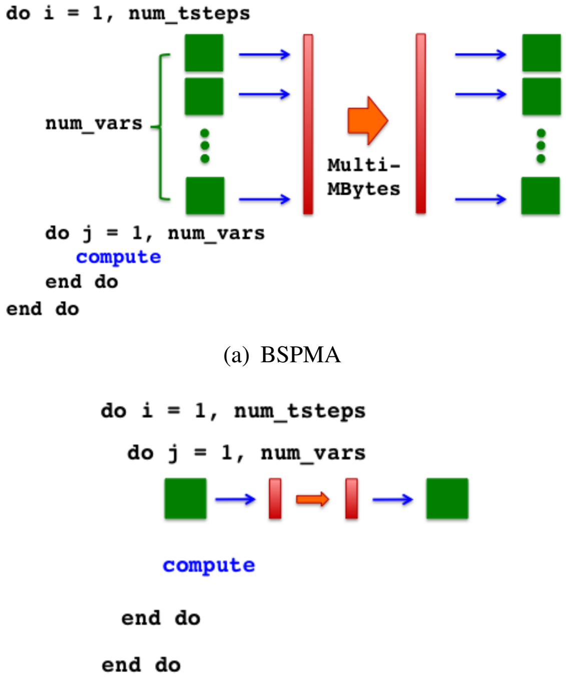


Fig. 2. CTH shaped charge simulation: time progresses left to right.



```
do i = 1, num_tsteps

    do j = 1, num_vars
        compute
    end do
end do
```

(a) BSPMA

```
do i = 1, num_tsteps
    do j = 1, num_vars

        compute

    end do
end do
```

(b) SVAF

Fig. 3. MiniGhost halo exchange strategies

large deformation, strong shock wave, solid mechanics code developed by Sandia National Laboratories [14]. The code solves the Lagrangian equations using second-order accurate numerical methods and mesh remap to reduce dispersion and dissipation including models for multi-phase, elastic viscoplastic, porous and explosive materials.

Studies involving several distinct problem sets show that their behavior, in terms of the boundary exchange, is consistent [16], allowing us to focus on one common configuration. The "shaped charge" problem [11], in three dimensions on a rectangular mesh, is illustrated in Figure 2.

The domain is divided into three dimensional regions which are mapped to parallel processes. Each MPI rank can have a maximum of six communication neighbors; for these problems that number is reached (for some ranks) once 128 ranks are employed. Boundary data (two dimensional "faces") is exchanged 19 times each time step. For the shaped charge problem set, each exchange aggregates data from 40 arrays, representing 40 variables, resulting in an average message size of 4.1 MBytes, illustrated in Figure 3(a).

Collective communication, called 90 times each time step, is typically a reduction (`MPI_Allreduce`) of small counts. Computation is characterized by regular memory accesses, is fairly cache friendly, with operations focusing on two dimensional planes.

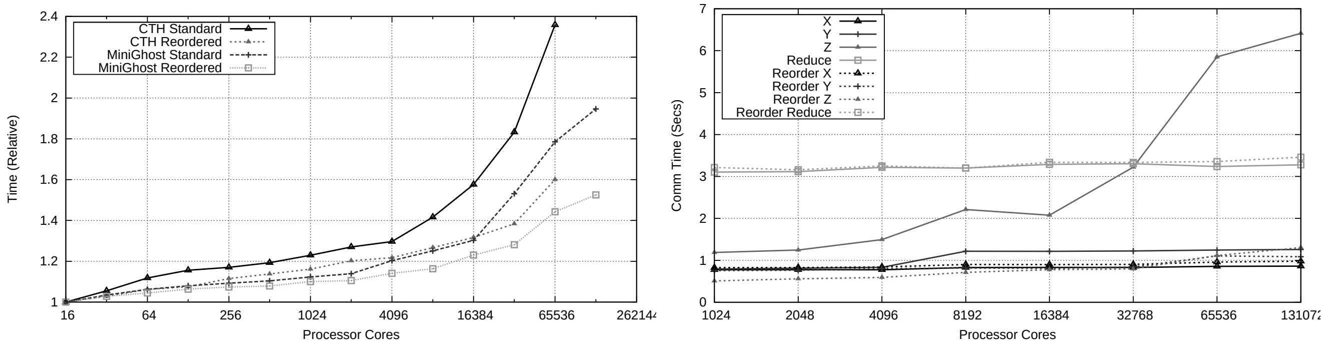
IV. Experiments

We begin with a study demonstrating the importance of mapping MPI ranks to the physical topology of the interconnect. Once effectively addressed, we explore an alternative communication strategy, designed for our exascale explorations.

A. *Mapping processes to processors*

CTH has performed well in a variety of computing environments over a long period of time [17], [19], [20]. However, a recent study involving high processor counts revealed significant scaling degradation in the nearest neighbor communication strategy employed in CTH [16]. The performance of a miniapp, called miniGhost [7] from the Mantevo project [13], configured to represent the communication requirements of CTH [6], began degrading at 4,096 processors, and was out of control by 131,072 processors (illustrated in Figure 4(a)). Note that the MPI reduction operations (typically global summation of a double precision scalar) scale well regardless of the process mapping, one indication of a quality MPI implementation.

The tractable nature of miniGhost enabled us to trace the problem to the communication in the z direction, illustrated in Figure 4(b). The problem was caused by the manner in which logical processes, organized as a logical three dimensional grid by CTH, were assigned to physical processors and mapped onto the torus. The combination of very large process counts and very large messages led to increasing contention. (We see similar behavior with the OpenMP+MPI implementation, where fewer but even larger messages are transmitted.)



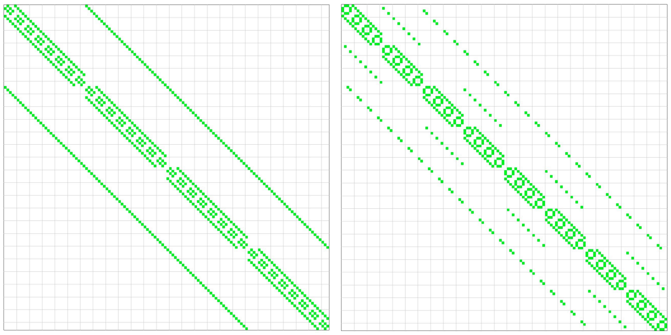
(a) Scaling

(b) Communication time

Fig. 4. CTH and miniGhost performance on Cielo

The number of hops (referred to as the *Manhattan distance*) required to communicate with x and y neighbors stays small as the number of processes increases, but the number in the z direction starts to grow rapidly after 2048 processes. Neighbors in the x direction required a maximum of one hop and in the y direction a maximum of two hops. This combined with the very large messages of a typical CTH problem set (e.g. for the "shaped charge" problem, 40 three dimensional state variable arrays generated message lengths of more than 5 MBytes) resulted in poor scaling beginning at 8,192 processes, a trend that accelerated after 16,384 processes.

In response, we implemented a means by which the parallel processes could be logically re-ordered to take advantage of the physical locality induced by the communication requirements. In the normal mode, CTH (and miniGhost) assigns blocks of the mesh to cores in a manner which ignores the connectivity of the cores in a node. On Cielo, as with other Cray X-series architectures, cores are numbered consecutively on a node, and this numbering continues on the next node. Blocks of the mesh are assigned to cores by traversing the blocks of the mesh in the x direction of the mesh starting at one corner of the mesh. Once those blocks are assigned, the next block assigned is the block one over in the y direction of the mesh from the first block assigned. The mesh is again then traversed in the x direction and blocks are assigned to cores. This process is continued until there are no more blocks in the y direction. The next block assigned is then the first block in the z direction from the first block assigned. The blocks of the mesh with this z value are then assigned as the first blocks were assigned. This process is then repeated until all blocks in the mesh have been assigned to cores in the machine.



(a) Original mapping

(b) Re-ordered mapping

Fig. 5. CTH logical processor mapping on Cielo

For example, the original approach logically maps 128 parallel processes onto a 4 × 8 × 4 grid of processes, illustrated in Figure 5(a).

The result is that the communication partners in the x direction are 1 away from the diagonal, those 4 away are the y neighbors, and those 32 away are the z neighbors. Our re-ordering algorithm assigns blocks of the mesh to the processor cores of the machine by groups. Illustrated in Figure 5(b), the re-ordered problem maps 2 × 2 × 4 blocks to each node. Here the communication partners that are 1 or 15 away are the x neighbors, those 2 or 30 away are the y neighbors, and those 30 away are the z neighbors. Since there are only four processor

Number of	Original Order			Re-ordered		
MPI ranks	X	Y	Z	X	Y	Z
16	0.0	0.0	0.0	0.0	0.0	0.0
32	0.0	0.0	0.0	0.0	0.0	0.0
64	0.0	0.0	0.3	0.0	0.3	0.0
128	0.0	0.0	1.0	0.0	0.5	0.0
256	0.0	0.0	1.0	0.0	0.5	0.3
512	0.0	0.1	2.0	0.0	0.6	0.4
1024	0.0	0.3	2.1	0.2	1.0	0.7
2048	0.0	0.3	2.7	0.3	1.2	1.2
4096	0.0	0.3	3.7	0.3	1.2	1.2
8192	0.0	0.5	5.1	0.2	1.1	2.0
16384	0.0	0.5	4.9	0.2	1.1	2.2
32768	0.0	0.5	5.6	0.2	1.1	2.5
65536	0.0	1.1	10.2	0.2	1.6	2.8
131072	0.0	1.1	10.1	0.2	1.6	3.1

TABLE II
MINIGHOST AVERAGE HOP COUNTS ON CIELO

	% off-node	
	Original	Re-ordered
x direction	3.23	48.39
y direction	100	49.21
z direction	100	23.81
Total	68.09	40.43

TABLE III
CTH OFF-NODE MESSAGE TRAFFIC ON CIELO

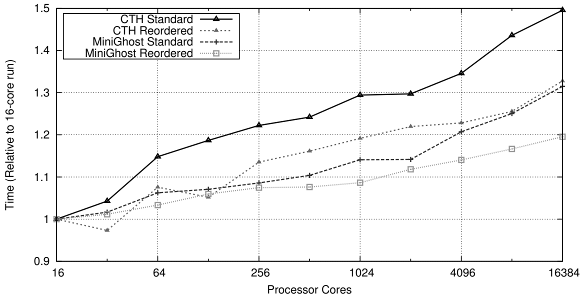


Fig. 6. CTH and miniGhost performance on on Chama

cores in the z direction and four processor cores in the z direction of the block on each node, there is only one z diagonal. In the case where there are more than four processor cores in the problem in the z direction, there would be another diagonal for z neighbors. The result is a slight increase in the average hop counts in the x and y directions, but a significant decrease in the average hop count in the z direction, as seen in Table II. The impact on scaling performance is evident in Figure 4. Communication time in the z direction is dramatically reduced, with only a slight increase in the x and y directions. The technique was applied to CTH, significantly improving its scaling characteristics as well. The remaining scaling issue is a function of synchronization costs, an issue to be more closely examined as we gain access to higher scales on Cascade systems.

Consider the largest CTH problem set run: $131,072$ processes, configured as a $32 \times 64 \times 64$ logical grid, each with a local domain of $160 \times 96 \times 80$, for a global domain of dimension $5120 \times 6144 \times 5120$. Over the course of execution, $770,048$ messages are transmitted, of size 2.57 Mbytes, 4.25 Mbytes, and 5.08 Mbytes in the x, y, and z directions, respectively, for a total volume of 3.06 TeraBytes of data. Table III shows the percentage of messages sent off-node, in each direction, for the original and re-ordered process execution.

At the expense of a 16-times increase in the number of off-node message in the x direction, traveling a short distance, the number and distance of off-node messages in the farther y and even farther z directions are reduced by more a factor of two and four, respectively. The number of stalls seen between the nodes and the NIC correlates with the hop count, indicating contention, but further analysis is ongoing in order to make stronger conclusions.

Similar experiments were performed on Chama, shown in Figure 6.

Again we see the benefit of the re-ordering scheme, as informed by miniGhost. What is not yet clear is the behavior that would be seen for significantly higher processor counts. However, the trends suggest that this approach is also viable for this architecture.

Export control restrictions on CTH preclude execution on Daint, but we are confident that miniGhost is sufficiently representative of CTH, as seen below.

B. Single variable boundary exchange

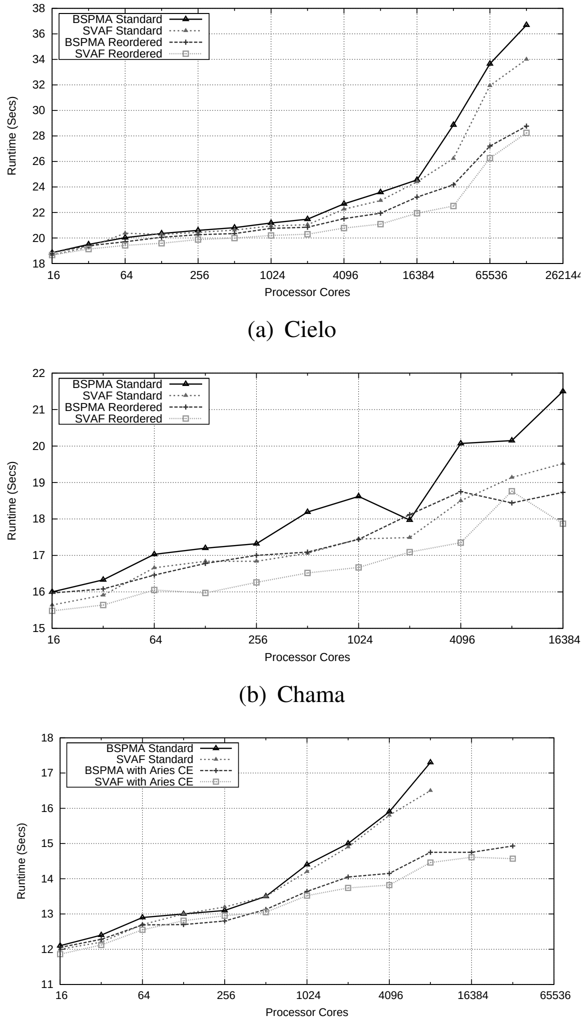
The BSPMA strategy adheres to an effective means for managing data movement. However, interconnect architects are telling us that exascale goals, especially with regard to power consumption, are causing significant changes in interconnects capabilities. In particular, injection rate and injection bandwidth are increasing proportionally more than global bandwidth. These changes are hinted at by

Cielo, providing an early opportunity to explore alternative strategies.

We configured miniGhost so that boundaries are exchanged as soon as computation on a variable is completed. Illustrated in Figure 3(b), we refer to this as “single variable aggregated faces” (SVAF). This means that, for the shaped charge problem, (up to) six messages, one to each neighbor, are transmitted for each variable update each time step.

This results in 39 more messages for each boundary exchange, each of which is 1/40 the size of the BSPMA messages. The expectation was that this would result in a significant degradation of overall performance on current architectures, but might hint at improvements on higher scales on future systems. Instead, as illustrated by the graph in Figure 7(a), the process re-ordered SVAF approach out-performed process re-ordered BSPMA.

Chama performance is shown in Figure 7(b). First, we note some apparent anomalies, which we attribute to system noise in spite of the dedicated environment. Further, the SVAF strategy is again more effective than BSPMA (excepting what appears to be an anomaly at 8,192 cores). Piz Daint results, illustrated in Figure 7(c), again show SVAF outperforming BSPMA, and perhaps hinting at better scaling with increasing core counts. Effective process mapping was used for each of the results shown. The Aries CE (“collective engine”) provides special support for reduction and barrier operations, here demonstrated to provide a critical service.



(a) Cielo

(b) Chama

(c) Cascade

Fig. 7. miniGhost BSPMA and SVAF Scaling

V. Summary

Expected architecture changes, driven by energy consumption and other constraints, compelled us to revisit some traditional and well-accepted strategies for sharing inter-process boundary data in PDE-based application programs. Herein we demonstrated an alternative to the ubiquitous BSPMA. Designed to exploit the capabilities of expected future architectures, the SVAF method provides a more natural coding style (data is sent as soon as it changes, maintaining stronger data coherence across the parallel processes), better use of available memory and bandwidth, and better scaling characteristics at high processor counts. Adding this strategy to CTH will require significant modification, so it is thus far untested in a full application. However, miniGhost has been demonstrated to represent CTH boundary exchange characteristics [6]. Its also worth noting that programming convenience has led some application developers (including those for CTH) to include the halo data for each variable in each exchange, often data is included that has not been modified. The SVAF strategy would eliminate this unnecessary data movement as the exchange is directly associated with the computation.

We also continue to see that in spite of improved interconnect technologies, process placement based on specific communication patterns is essential to exploiting the capabilities of the architecture. Similar experiments on different architectures, including a 6-core based XE6 (Hopper at NERSC) as well as the host-device configuration of the Cray XK6, also show benefits of this work, to be presented in future publications. Experiments using OpenMP for on-node computation reduces the overall message traffic, but the resulting messages are larger and thus also cause contention that adversely impacts scaling performance.

Future work includes a miniGhost implementation designed to hide communication costs through explicit computational overlap strategies, using MPI functionality as well as Fortran co-arrays. More complex computations are being incorporated to represent a broader set of codes, including those based on AMR. Our work is also informing the development of a more general process mapping capability [10], useful to a broader set of applications.

Acknowledgment

We thank the Swiss National Supercomputer Service (CSCS) for early access to their Cascade System Piz Daint. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] S. Ahern et al. Scientific Application Requirements for Leadership Computing at the Exascale. Technical Report TM-2007/238, Oak Ridge National Laboratory, December 2007.

[2] S. Alam et al. Early evaluation of IBM BlueGene/P. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing (SC'08)*, SC'08, pages 23:1–23:12, Piscataway, NJ, USA, 2008. IEEE Press.

[3] B. Alverson, E. Froese, L. Kaplan, and D. Roweth. Cray XC Series Network. Technical Report WP-Aries01-1112, Cray Inc., 2012.

[4] D.H. Bailey, E. Barszcz, J.T. Barton, D.S. Browning, R.L. Carter, D. Dagum, R.A. Fatoohi, P.O. Frederickson, T.A. Lasinski, R.S. Schreiber, H.D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The NAS Parallel Benchmarks. *International Journal of Supercomputer Applications*, 5(3):63–73, Fall 1991.

[5] R.F. Barrett, S. Ahern, M.R. Fahey, R. Hartman-Baker, J.K. Horner, S.W. Poole, and R. Sankaran. A Taxonomy of MPI-Oriented Usage Models in Parallelized Scientific Codes. In *The International Conference on Software Engineering Research and Practice*, 2009.

[6] R.F. Barrett, P.S. Crozier, D.W. Doerfler, S.D. Hammond, M.A. Heroux, P.T. Lin, H.K. Thornquist, T.G. Trucano, and C.T. Vaughan. Summary of Work for ASC L2 Milestone 4465: Characterize the Role of the Mini-Application in Predicting Key Performance Characteristics of Real Applications. Technical Report SAND2012-4667, Sandia National Laboratories, 2012.

[7] R.F. Barrett, C.T. Vaughan, and M.A. Heroux. MiniGhost: A Miniapp for Exploring Boundary Exchange Strategies Using Stencil Computations in Scientific Parallel Computing. Technical Report SAND2011-5294832, Sandia National Laboratories, May 2011.

[8] B. Bland. Jaguar: The World's Most Powerful Computer System. In *The 52nd Cray User Group meeting*, 2010.

[9] L. Brown, P.M. Bennett, M. Cowan, C. Leach, and T.C. Oppe. Finding the Best HPCMP Architectures Using Benchmark Application Results for TI-09. *HPCMP Users Group Conference*, 0:416–421, 2009.

[10] S.P. Feer, Z.D. Rhodes, N.W. Price, D.P. Bunde, and V.J. Leung. Task Mapping for Noncontiguous Allocations. SAND 2011-7334C; Under review, 2013.

[11] D.R. Gardner and C.T. Vaughan. The Optimization of a Shaped-Charge Design Using Parallel Computers. Technical Report SAND99-2953, Sandia National Laboratories, 1999.

[12] G. Hager, G. Jost, and R. Rabenseifner. Communication Characteristics and Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-core SMP Nodes. In *Proc. 51st Cray User Group Meeting*, 2009.

[13] M.A. Heroux, D.W. Doerfler, P.S. Crozier, J.M. Willenbring, H.C. Edwards, A. Williams, M. Rajan, E.R. Keiter, H.K. Thornquist, and R.W. Numrich. Improving Performance via Mini-applications. Technical Report SAND2009-5574, Sandia National Laboratories, September 2009.

[14] E.S. Hertel, Jr., R. L. Bell, M. G. Elrick, A. V. Farnsworth, G. I. Kerley, J. M. Mcglaun, S. V. Petney, S. A. Silling, P. A. Taylor, and L. Yarrington. CTH: A Software Family for Multi-Dimensional Shock Physics Analysis. In *Proceedings, 19th International Symposium on Shock Waves*, pages 377–382, 1993.

[15] T. Hoefler et al. The Scalable Process Topology Interface of MPI 2.2. *Concurr. Comput. : Pract. Exper.*, 23(4):293–310, March 2011.

[16] S.M. Kelly et al. Report of Experiments and Evidence for ASC L2 Milestone 4467 - Demonstration of a Legacy Application's Path to Exascale. Technical Report SAND2012-1750, Sandia National Laboratories, 2012.

[17] T.M. Kendall and S.J. Schraml. Early Experiences with the IBM POWER4 Scalable Parallel System at the ARL MSRC. In *Proceedings of the DOD Higher Performance Computing User Group Conference*, June 2002. ARL-RP-281.

[18] J. Kim, W. J. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable dragonfly topology. *SIGARCH Comput. Archit. News*, 36(3):77–88, June 2008.

[19] M. Rajan, C.T. Vaughan, D.W. Doerfler, R.F. Barrett, P.T. Lin, K.T. Pedretti, and K.S. Hemmert. Application-driven analysis of two generations of capability computing: the transition to multicore processors. *Concurr. Comput. : Pract. Exper.*, 24(18):2404–2420, December 2012.

[20] S.J. Schraml and T.M. Kendall. Scalability of the CTH Shock Physics Code on the Cray XT. In *Proceedings of the DOD Higher Performance Computing User Group Conference*, June 2009. ARL-RP-281.

[21] H. Simon, T. Zacharia, and R. Stevens. Modeling and Simulation at the Exascale for Energy and the Environment: Report on the Advanced Scientific Computing Research Town Hall Meetings on Simulation and Modeling at the Exascale for Energy, Ecological Sustainability and Global Security (E3). Technical report, Office of Science, The U.S. Department of Energy, 2007.

[22] L.G. Valiant. A Bridging Model for Parallel Computation. *Commun. ACM*, 33:103–111, August 1990.

[23] C.T. Vaughan. Application Characteristics and Performance on a Cray XE6. In *Proc. 53th Cray User Group Meeting*, 2011.

[24] H. Yu, I.H. Chung, and J. Moreira. Topology Mapping for Blue Gene/L Supercomputer. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.